Efficient Representation of Timed UML 2 Interactions

Knapp, Alexander; Störrle, Harald

Published in:
Proceedings of the 8th International Conference on System Analysis and Modeling

Link to article, DOI:
10.1007/978-3-319-11743-0_8

Publication date:
2014

Document Version
Peer reviewed version

Efficient Representation of Timed UML 2 Interactions

Alexander Knapp[1] and Harald Störrle[2]

[1] Universität Augsburg, Germany
knapp@informatik.uni-augsburg.de
[2] Danmarks Tekniske Universitet, Denmark
hsto@dtu.dk

Abstract. UML 2 interactions describe system behavior over time in a declarative way. The standard approach to defining their formal semantics enumerates traces of events; other representation formats, like Büchi automata or prime event structures, have been suggested, too. We describe another, more succinct format, interaction structures, which is based on asymmetric event structures. It simplifies the integration of real time, and complex operators like alt and break, and leads to an efficient semantic representation of interactions. We provide the formalism, and a prototypical implementation highlighting the benefits of our approach.

1 Introduction

Among the many languages defined in UML 2, interactions are among the most widely used [2,3]. They describe system behavior over time in a declarative way, focusing on the message exchange between instances. Thus, interactions are well-suited to specify temporal constraints. A sample UML 2 interaction is shown in Fig. 1 below.

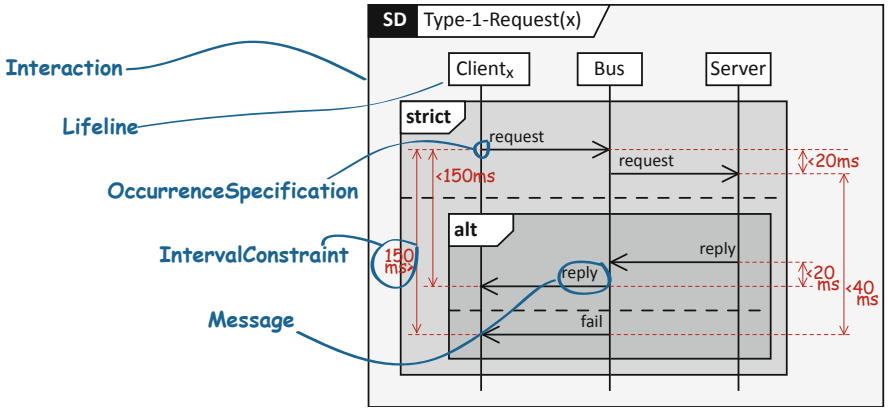


Fig. 1. A first example of a UML 2 interaction. Time constraints are highlighted in red, explanations of UML concepts are shown blue.

Equipped with a suitable formal semantics, UML 2 interactions can be used for rigorous analysis of system specifications, in particular, checking the consistency between

D. Amyot et al. (Eds.): SAM 2014, LNCS 8769, pp. 110–125, 2014.

different parts of a specification. In the context of run-time verification and online checking, it is particularly interesting whether a given interaction specifying a system's behavior is temporally sound. More precisely, are the temporal constraints logically consistent? Do they hold for a given trace? Do they hold for all traces?

Existing formal semantics for UML 2 interactions such as [14] already allow to answer such questions, though only in theory: simply compute the set of all traces and check for emptiness of the set (logical consistency) or inclusion of the target trace. So, clearly, we can decide the above consistency questions for all finite traces in principle. However, due to the existence of interaction operators like par, the number of traces is exponential in the size of the interaction, so we would have to enumerate a very large set of traces before we can answer the questions raised above. Clearly, this approach is of little practical value.

Unfortunately, all existing semantics that include real time and the interaction operators (which distinguish UML 2 from UML 1 interactions) suffer from this limitation (or use an exponentially sized semantic representation of the interaction to begin with). In contrast, the approach presented in this paper introduces a novel semantic representation that represents the set of all traces of an interaction in a format that grows linearly in the size of the underlying interaction. It allows to check whether a concrete run complies with an interaction and its temporal constraints, and it can be implemented efficiently, with very modest effort. This is reminiscent of the way binary decision diagrams (BDDs) improved model checking of propositional logic formulas.

Synopsis. After discussing related work in Sect. 2, we summarize the sub-language of UML 2 interactions that we consider in Sect. 3. Our format of symbolic representations of these interactions is introduced in Sect. 4, where we also discuss the resulting traces and the translation of UML 2 interactions into a symbolic representation. In Sect. 5, we give an overview of our prototypical implementation and its performance. We conclude and discuss future work in Sect. 6.

2 Related Work

A comprehensive survey of UML 2 interaction semantics is found in [9]. Among other things, the transition from UML 1 to UML 2 introduced a novel semantics for interactions, which is why the first investigations of UML 2 interactions [14,13] focused on understanding and interpreting the standard document. Since the UML specification informally suggests that the meaning of interactions are sets of sequences of so-called "interaction occurrences", this is what the first semantics defined formally.

While this point of view was well-suited to understand and formalize the prose specification of UML 2, it is less-well suited for the analysis of interactions and their possible traces, since interactions give rise to an exponential number of traces. The same problem is encountered by approaches using an automata-based representation (e.g., [7]), as they need to encode traces in states, which again leads to an exponential number of states. The declarative representation we propose in this paper, on the other hand, encodes UML 2 interaction as sets of constraints whose size is linear in the size of the interaction. Furthermore, it allows to include real-time annotations seamlessly. It

is suitable for checking whether a given trace of time-stamped events is a valid trace according to the interaction without having to compute all its possible traces.

An approach similar to ours has first been pursued by Küster-Filipe [8]. While Küster-Filipe employed prime event structures, we propose to use a format inspired by asymmetric event structures [1], which yields a more compact symbolic representation by avoiding duplications that are required when using prime event structures. At the same time, asymmetric event structures also allow us to integrate UML 2's break operator for breaking scenarios, an important practical scenario not covered by Küster-Filipe.

There is a rich body of work on timed Message Sequence Diagrams and timed UML 1 interactions, but there are only three approaches to study timed UML 2 interactions according to [9]. None of these has been implemented; in contrast, we do present a prototypical implementation for checking the conformance of timed traces w.r.t. an interaction, and discuss its performance.

There have also been other approaches that focus on different aspects of interactions. For model checking against the automaton-based specification format of UML 2 state machines, a representation of an interaction as an additional observer Büchi automaton is a closer fit [7]; also, the relation of interactions to safety and liveness properties can be expressed when using Büchi automata [4]. For testing, a representation as a structured composite graph makes the decision structure more transparent, which can be used to derive test data [10]. For studying the concurrency inherent in an interaction, the use of (prime) event structures, a denotational framework for true concurrency, [8] or lattices [5] turned out to be fruitful.

3 UML 2 Interactions

The main building block is the basic interaction, which represents orderings of so-called "occurrence specifications" directly, as a partial order. An occurrence specification captures that an event (like the sending or receiving of a message) happens on an instance partaking in the interaction; the ordering relations define the sequences in which these events may happen. Basic interactions form an "interaction fragment" that may be combined by the "interaction operators" strict (strict sequential composition), seq (lifeline-wise sequential composition), par (parallel composition), alt (alternative composition), break (aborting composition), and ref (including a named interaction). Additionally, all fragments may be equipped with timing constraints. For later reuse by ref, interaction fragments can be given a name.

We assume three primitive, finite domains for *instances* $\mathcal{I}$, *messages* $\mathcal{M}$, and interaction *names* $\mathcal{N}$. We always assume that all identifiers of occurrence specifications are globally unique. An *occurrence specification* is of the form $o : \tau$, where o is the identifier of the occurrence specification and τ is its *type*. The type of an occurrence specification is of one of the forms $\text{SND}(s, r, m)$ or $\text{RCV}(s, r, m)$, representing the dispatch and the arrival of message m from *sender* instance s to *receiver* instance r, respectively. The set $\mathcal{O}$ comprises all occurrence specifications over $\mathcal{I}$ and $\mathcal{M}$. For an $o : \tau \in \mathcal{O}$, we write $\tau(o)$ for τ.

A *basic interaction* B is given by a directed acyclic graph $(O, \rightarrow)$ with $O \neq \emptyset$ and $O \subseteq \mathcal{O}$ a finite, non-empty set of occurrence specifications such that the identifiers of

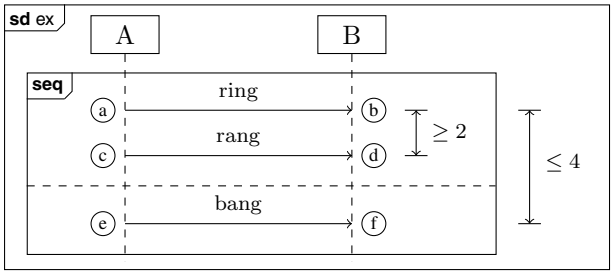
the occurrence specifications in O are all different, and $\rightarrow \subseteq O \times O$ such that the reflexive-transitive closure of $\rightarrow$ on O forms a partial order. The abstract syntax of our fragment of UML 2 interactions is given by the grammar in Fig. 2.

$$
\begin{array}{rcl}
\textit{TimingConstraint} \ni \Gamma & ::= & o_2 - o_1 \bowtie d \mid \ell \bowtie d \\
& & \mid \mathrm{true} \mid \Gamma_1 \wedge \Gamma_2 \mid \Gamma_1 \vee \Gamma_2 \\
\textit{Interaction} \ni I & ::= & \mathsf{sd}(name, T) \\
\textit{InteractionFragment} \ni T & ::= & B \mid CF \mid \mathsf{tmconstr}(T, \Gamma) \\
\textit{CombinedFragment} \ni CF & ::= & \mathsf{strict}(T_1, T_2) \mid \mathsf{seq}(T_1, T_2) \mid \mathsf{par}(T_1, T_2) \\
& & \mid \mathsf{alt}(T_1, T_2) \mid \mathsf{break}(T_1, T_2) \mid \mathsf{ref}(name)
\end{array}
$$

Fig. 2. Abstract syntax of timed interactions: $o_1, o_2 \in \mathcal{O}$, $\bowtie \in \{<, \leq, \geq, >\}$, and $d \in \mathbb{Q}_{\geq 0}$; B ranges over the basic interactions, $name$ over the interaction names $\mathcal{N}$

For expressing timing constraints, we use clauses of the form Γ specified in Fig. 2. Intuitively, a timing constraint $o_2 - o_1 \bowtie d$ means that the difference in time between any occurrence of an event conforming to o_2 and any event conforming to o_1 is bounded by d w.r.t. to the relation $\bowtie$. A timing constraint $\ell \bowtie d$ means that the duration of the interaction fragment to which this timing constraint is attached is bounded by d w.r.t. to $\bowtie$. Furthermore, true represents the timing constraint that is always true, and $\Gamma_1 \wedge \Gamma_2$ and $\Gamma_1 \vee \Gamma_2$ respectively mean the conjunctive and disjunctive combination of the timing constraints Γ_1 and Γ_2. Though we have restricted ourselves to binary relations over occurrence specifications, this language can be extended easily for correlating an arbitrary number of occurrence specifications.

Example 1. Consider the following UML 2 interaction diagram:



In the abstract syntax, this is represented by $\mathsf{sd}(\mathsf{ex}, \mathsf{tmconstr}(\mathsf{seq}(\mathsf{tmconstr}(\mathsf{B}_1, \mathsf{T}_1), \mathsf{B}_2), \mathsf{T}_2))$ with basic interaction B_1 and B_2, and timing constraints T_1 and T_2 given by

$$
\begin{aligned}
\mathsf{B}_1 = (&\{ⓐ : \textsc{snd}(\mathrm{A}, \mathrm{B}, \mathrm{ring}), ⓑ : \textsc{rcv}(\mathrm{A}, \mathrm{B}, \mathrm{ring}), \\
&\ ⓒ : \textsc{snd}(\mathrm{A}, \mathrm{B}, \mathrm{rang}), ⓓ : \textsc{rcv}(\mathrm{A}, \mathrm{B}, \mathrm{rang})\}, \\
&\{ⓐ \rightarrow ⓑ, ⓒ \rightarrow ⓓ, ⓐ \rightarrow ⓒ, ⓑ \rightarrow ⓓ\}) , \\
\mathsf{B}_2 = (&\{ⓔ : \textsc{snd}(\mathrm{A}, \mathrm{B}, \mathrm{bang}), ⓕ : \textsc{rcv}(\mathrm{A}, \mathrm{B}, \mathrm{bang})\}, \{ⓔ \rightarrow ⓕ\}) , \\
\mathsf{T}_1 = &\ ⓓ - ⓑ \geq 2 , \\
\mathsf{T}_2 = &\ ⓓ - ⓑ \leq 4 .
\end{aligned}
$$

□

The language of interactions can be extended by introducing syntactical abbreviations like, e.g., a finite upper-bounded $\mathsf{loop}(k, T)$ setting

$$\mathsf{loop}(1, T) \equiv T$$
$$\mathsf{loop}(k+1, T) \equiv \mathsf{alt}(T, \mathsf{seq}(T\rho_1, \mathsf{loop}(k, T\rho_2)))$$

where the renamings ρ_1 with $\rho_1(o) = \mathsf{seq}.o$ and ρ_2 with $\rho_2(o) = \mathsf{loop}.o$, written in postfix notation, introduce consistently new names for the occurrence specifications of T.

4 Symbolic Representation of UML 2 Interactions

According to the UML 2 standard, (timed) UML 2 interactions describe "emergent behavior" [12], i.e., UML 2 interactions can be considered to specify which traces of events are allowed (or disallowed) to be observed from an implemented system. The occurrence specifications of a UML 2 interaction express the possible events. On the one hand, the orderings of these occurrence specifications, be they directly given by a basic interaction, or be they expressed by the interaction operators strict, seq, or par, restrict the possible sequences of events. On the other hand, compositions of interaction fragments via alt and break specify choices between different sequences. Usages of ref merely correspond to macro expansions. Finally, the timing constraints restrict timing distances between events for the occurrence specifications.

Formally, a (timed) *event* $e = \langle \tau, t \rangle$ consists of two parts: the type τ of an occurrence specification, saying whether it is a sending or receiving event, for which message, and between which instances; and the time point $t \in \mathbb{R}_{\geq 0}$ at which it occurs. We write $\tau(e)$ for τ, and $t(e)$ for t. We say that an event e *conforms to* an occurrence specification o, if $\tau(e) = \tau(o)$. A sequence of events $e_1\ e_2\ \ldots\ e_k$ is a *trace* if $t(e_1) \leq t(e_2) \leq \ldots \leq t(e_k)$.

We now want to capture the prescriptions mandated by a UML 2 interaction in a symbolic format that succinctly expresses the requirements on what the allowed traces of (timed) events are. We call this format an *interaction structure* of a UML 2 interaction. Such an interaction structure (O, R, X, Θ) consists of the following components:

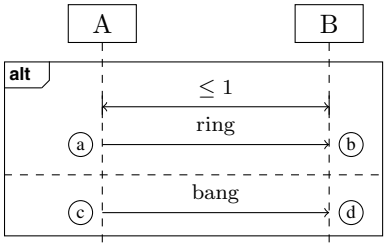
- a finite set of occurrence specifications O; it specifies all the occurrence specifications for which events are allowed to be observed: in a trace $e_1\ \ldots\ e_k$, all events e_i have to conform to one of the occurrence specifications in O. However, there may be several choices for conformance and we have to provide an injective map $\lambda : \{e_1, \ldots, e_k\} \to O$ with $\tau(e) = \tau(\lambda(e))$ for all $e \in \{e_1, \ldots, e_k\}$ in order to fix which event represents which occurrence specification.
- a binary relation $R \subseteq O \times O$ specifying a *causality relation* over O, i.e., a partial ordering on O. This relation says in which order the events conforming to O are allowed to occur, if they occur at all: if for a trace $e_1\ \ldots\ e_k$ the events e_i and e_j with $i \leq j$ shall represent the occurrence specifications o_i and o_j in O, then it must not be the case that $o_j \preceq_R o_i$ in the partial order $\preceq_R$ generated by R.
- a binary relation $X \subseteq O \times O$ specifying an R-compatible *inhibition relation* over O, i.e., an irreflexive relation $\rhd_{(R,X)} \subseteq O \times O$ with $o_2 \rhd_{(R,X)} o_3$ iff there is an $o_1 \in O$

with $o_1 \preceq_R o_2$ and $(o_1, o_3) \in X$. This relation expresses which events inhibit others: if $o_1 \rhd_{(R,X)} o_2$, then an event e representing the occurrence specification o_1 excludes events conforming to o_2 from occurring after e in a trace.

– a timing constraint Θ, which is a conjunctive or disjunctive combination of timing constraints of the form true or $o_2 - o_1 \bowtie d$; it says which timing conditions the time-stamps of events have to obey (where duration constraints of the form $\ell \bowtie d$ are reduced to combinations of occurrence constraints).

For the traces of an interaction structure (O, R, X, Θ), we require that before each event on the trace a maximal, consistent set of events w.r.t. to R, X, and Θ occurs: All the causes of the event w.r.t. the causality relation $\preceq_R$ which are not present on the trace have to be excluded by the inhibition relation $\rhd_{(R,X)}$, and all timing constraints from Θ have to be satisfied for the chosen events; see Sect. 4.1.

Example 2. Consider the following UML 2 interaction diagram:



The traces of events allowed by this interaction are of one of the following two forms:

$$\langle \textsc{snd}(\mathrm{A}, \mathrm{B}, \mathrm{ring}), t_1 \rangle \, \langle \textsc{rcv}(\mathrm{A}, \mathrm{B}, \mathrm{ring}), t_2 \rangle, \quad t_1, t_2 \in \mathbb{R}_{\geq 0}, \quad t_2 - t_1 \leq 1 \, ;$$
$$\langle \textsc{snd}(\mathrm{A}, \mathrm{B}, \mathrm{bang}), t_3 \rangle \, \langle \textsc{rcv}(\mathrm{A}, \mathrm{B}, \mathrm{bang}), t_4 \rangle, \quad t_3, t_4 \in \mathbb{R}_{\geq 0} \, .$$

The requirements on the occurrence specifications themselves are: ⓐ is ordered before ⓑ, and ⓒ before ⓓ; either the upper operand can be observed, i.e., ⓐ and ⓑ occur, or the lower operand can be observed, i.e., ⓒ and ⓓ occur; at most a single time unit elapses between ⓐ and ⓑ. We can express these requirements by the following interaction structure:

$$\begin{aligned}
O &= \{ⓐ, ⓑ, ⓒ, ⓓ\} \, , \\
R &= \{ⓐ \to ⓑ, ⓒ \to ⓓ\} \, , \\
X &= \{ⓐ \rightsquigarrow ⓒ, ⓐ \rightsquigarrow ⓓ, ⓑ \rightsquigarrow ⓒ, ⓑ \rightsquigarrow ⓓ, \\
&\qquad ⓒ \rightsquigarrow ⓐ, ⓓ \rightsquigarrow ⓐ, ⓒ \rightsquigarrow ⓑ, ⓓ \rightsquigarrow ⓑ\} \, , \\
\Theta &= ⓑ - ⓐ \leq 1 \, ,
\end{aligned}$$

where we write $o \rightsquigarrow o'$ for a pair $(o, o') \in X$. Relation R requires that ⓐ only may be observed before ⓑ, and that ⓒ may only be observed before ⓓ. The interpretation of X is that an observation of ⓐ or ⓑ must not be followed by an observation of ⓒ or ⓓ in the future, and, symmetrically, that the observation of ⓒ or ⓓ must not be followed by an observation of ⓐ or ⓑ in the future. In combination, R and X say that

an observation of ⓑ not only cannot be followed by an observation of ⓒ or ⓓ but must be preceded by an observation of ⓐ, since ⓐ $\rightarrow$ ⓑ, and not ⓐ $\rightsquigarrow$ ⓑ.

Thus the following non-empty sequences of occurrence specifications conform to both the ordering constraints R and the inhibition constraints X:

ⓐ, ⓐⓑ, ⓒ, ⓒⓓ.

Generally, we would require that either the upper or the lower operand are observed completely, which then only leaves ⓐⓑ and ⓒⓓ.

For taking into account the timing constraints, we look for all traces of events for which we can find a bijective labeling from the set of events in the trace to a sequence of occurrence specifications conforming to the interaction structure such that the concrete time-stamps of the events satisfy the conditions of the timing constraints. For ⓐⓑ this results in

$$\langle \textsc{snd}(\mathrm{A}, \mathrm{B}, \mathrm{ring}), t_1 \rangle \, \langle \textsc{rcv}(\mathrm{A}, \mathrm{B}, \mathrm{ring}), t_2 \rangle, \quad t_1, t_2 \in \mathbb{R}_{\geq 0}, \quad t_2 - t_1 \leq 1$$

using the labeling $\lambda(\langle \textsc{snd}(\mathrm{A}, \mathrm{B}, \mathrm{ring}), t_1 \rangle) =$ ⓐ (both the event and the occurrence specification have the same type), and $\lambda(\langle \textsc{rcv}(\mathrm{A}, \mathrm{B}, \mathrm{ring}), t_2 \rangle) =$ ⓑ. Similarly, ⓒⓓ yields

$$\langle \textsc{snd}(\mathrm{A}, \mathrm{B}, \mathrm{bang}), t_3 \rangle \, \langle \textsc{rcv}(\mathrm{A}, \mathrm{B}, \mathrm{bang}), t_4 \rangle, \quad t_1, t_2 \in \mathbb{R}_{\geq 0}$$

using the labeling $\lambda(\langle \textsc{snd}(\mathrm{A}, \mathrm{B}, \mathrm{bang}), t_3 \rangle) =$ ⓒ and $\lambda(\langle \textsc{rcv}(\mathrm{A}, \mathrm{B}, \mathrm{bang}), t_4 \rangle) =$ ⓓ; here the timing constraint $\Theta =$ ⓑ $-$ ⓐ ≤ 1 is satisfied, since the labeling does not mention ⓐ and ⓑ. □

For an interaction structure $S = (O, R, X, \Gamma)$, we write $O(S)$, $R(S)$, $X(S)$, and $\Gamma(S)$ for O, R, X, and Γ, respectively.

The format of interaction structures is inspired by the notion of *prime event structures* $(E, \leq, \sharp)$, where E is a set of *events*, $\leq \subseteq E \times E$ is a partial order describing the *causal relationship* of events, and $\sharp$ is an irreflexive, symmetric binary relation $\sharp \subseteq E \times E$, specifying which events are in *conflict* with each other [11]. In a *configuration* $C \subseteq E$ of the prime event structure, all causes of each event have to be present and any two events must not be in conflict.

Küster-Filipe [8] has suggested to capture a UML 2 interaction as a prime event structure, where its (partial) executions correspond to the configurations of the prime event structure. However, when expressing alt by the symmetric conflict relation of a prime event structure, it is necessary to duplicate all future events: Consider strict(alt(ⓐ, ⓑ), T), where ⓐ and ⓑ represent basic interactions of a single occurrence specification; here, ⓐ and ⓑ are in conflict. If all occurrence specifications of the interaction fragment T get ⓐ and ⓑ as their causes, no configuration containing ⓐ or ⓑ could also contain any occurrence specification from T, since then also all the causes of this occurrence specification, which are both ⓐ and ⓑ, would have to be present, which is impossible. Thus, the occurrence specifications of T are duplicated and one copy gets only ⓐ as its cause, the other copy ⓑ.

We circumvent this duplication process by using the notion of *asymmetric* conflicts taken from *asymmetric event structures* [1]. There, conflicts are expressed by *weak causes* saying that if an event e is a weak cause for another event e' and both events e and e' occur in a configuration then e has to precede e'; in fact, if e' also would be a weak cause for e, then e and e' could not occur simultaneously in one configuration. In our approach, we rely exclusively on such weak causes, i.e., both R and X of an interaction structure (O, R, X, Θ) are interpreted in this way. This makes the presentation more uniform, though at the expense of requiring that all possible weak causes of an occurrence specification have to be present in a trace.

4.1 Traces of an Interaction Structure

For an interaction structure (O, R, X, Θ), we now define its traces of events following the recipe of the last example. We proceed in two steps: First, we define all sequences of occurrence specifications (not events) that are allowed by the interaction structure. Then we take the timing constraints into account and define the traces of (O, R, X, Θ).

For the first step, let $o_1 \ldots o_k$ be a sequence of different occurrence specifications with $\{o_1, \ldots, o_k\} \subseteq O$. Let $\preceq_R$ be the partial order relation generated by R through taking the reflexive, transitive closure of R on O, and let $\rhd_{(R,X)}$ be the inhibition relation generated from R and X by taking the upwards closure of X w.r.t. $\preceq_R$. We say that $o_1 \ldots o_k$ *conforms to* the ordering constraints R and the inhibition constraints X if for all $1 \leq j \leq k$ the occurrence specification o_j is a minimal element of the partial order $(O_j, \preceq_R \cap (O_j \times O_j))$ with

$$O_j = O \setminus (\{o_1, \ldots, o_{j-1}\} \cup \{o \in O \mid \exists 1 \leq i \leq j-1 \,.\, o_i \rhd_{(R,X)} o\}) \,.$$

The sequence of occurrence specifications $o_1 \ldots o_k$ is *allowed by* (O, R, X, Θ) if it conforms to R and X and is maximal w.r.t. conformance, i.e., there is no $o \in O \setminus \{o_1, \ldots, o_k\}$ such that also $o_1 \ldots o_k\, o$ conforms to R and X.

Example 3. Consider the following interaction structure (O, R, X, true) (where we omit the occurrence specification types):

$$O = \{ⓐ, ⓑ, ⓒ, ⓓ\} \,,$$
$$R = \{ⓐ \to ⓑ, ⓒ \to ⓓ\} \,,$$
$$X = \{ⓒ \rightsquigarrow ⓐ, ⓒ \rightsquigarrow ⓑ, ⓓ \rightsquigarrow ⓐ, ⓓ \rightsquigarrow ⓑ\} \,.$$

Here, every sequence of occurrence specifications allowed by (O, R, X, Θ) must not show ⓐ or ⓑ after ⓒ or ⓓ. On the other hand, ⓑ ⓒ is not allowed, since ⓐ $\to$ ⓑ, i.e., ⓑ is not a minimal element of $\preceq_R$. By the maximality condition, the allowed sequences of occurrence specifications are:

ⓒ ⓓ , ⓐ ⓒ ⓓ , ⓐ ⓑ ⓒ ⓓ . □

For the second step, taking the timing constraints Θ into account, let $e_1 \ldots e_k$ be a trace of events. We say that $e_1 \ldots e_k$ *conforms to* a sequence of occurrence specifications $o_1 \ldots o_l$ allowed by (O, R, X, Θ) *via* a function $\lambda : \{e_1, \ldots, e_k\} \to$

$\{o_1, \ldots, o_l\}$ if λ is bijective and $\tau(e) = \tau(\lambda(e))$ for all $e \in \{e_1, \ldots, e_n\}$; we call such a function a *labeling*. Now, let $e_1 \ldots e_k$ conform to $o_1 \ldots o_l$ via the labeling λ. The trace of events $e_1 \ldots e_k$ *satisfies* a time constraint $o'_2 - o'_1 \bowtie d$ w.r.t. λ if either $\{\lambda(e) \mid e \in \{e_1, \ldots, e_k\}\} \neq \{o'_1, o'_2\}$, i.e., at least one of the occurrence specifications mentioned by the timing constraint is not covered by the trace of events; or if $t(\lambda^{-1}(o'_2)) - t(\lambda^{-1}(o'_1)) \bowtie d$, i.e., the time difference between the events representing o'_2 and o'_1, respectively, is bounded by d w.r.t. $\bowtie$ (with its usual meaning on the real numbers). The trace *satisfies* a time constraint $\Theta_1 \wedge \Theta_2$ w.r.t. λ if it satisfies both Θ_1 and Θ_2 w.r.t. λ; and it *satisfies* a time constraint $\Theta_1 \vee \Theta_2$ w.r.t. λ if it satisfies Θ_1 or Θ_2 w.r.t. λ.

Summing up, a trace of events $e_1 \ldots e_k$ *satisfies* the interaction structure (O, R, X, Θ) if there is a sequence of occurrence specifications $o_1 \ldots o_l$ allowed by (O, R, X, Θ) such that $e_1 \ldots e_k$ conforms to $o_1 \ldots o_l$ via a labeling $\lambda : \{e_1, \ldots, e_k\} \to \{o_1, \ldots, o_l\}$ and $e_1 \ldots e_k$ satisfies Θ w.r.t. λ.

Example 4. Consider the interaction structure of the previous example where now the occurrence specification types are

$$\text{ⓐ} : \textsc{snd}(\mathrm{A}, \mathrm{B}, \mathrm{ring}) \,, \quad \text{ⓑ} : \textsc{rcv}(\mathrm{A}, \mathrm{B}, \mathrm{ring}) \,,$$
$$\text{ⓒ} : \textsc{snd}(\mathrm{A}, \mathrm{B}, \mathrm{ring}) \,, \quad \text{ⓓ} : \textsc{rcv}(\mathrm{A}, \mathrm{B}, \mathrm{ring}) \,;$$

and where we replace the timing constraint true by $\text{ⓑ} - \text{ⓐ} \leq 1$. Then the trace of events $\langle \textsc{snd}(\mathrm{A}, \mathrm{B}, \mathrm{ring}), 0.2\rangle \langle \textsc{rcv}(\mathrm{A}, \mathrm{B}, \mathrm{ring}), 1.3\rangle$ satisfies this interaction structure, since it conforms to the sequence of occurrence specification ⓒⓓ via the labeling $\lambda(\langle \textsc{snd}(\mathrm{A}, \mathrm{B}, \mathrm{ring}), 0.2\rangle) = \text{ⓒ}$ and $\lambda(\langle \textsc{rcv}(\mathrm{A}, \mathrm{B}, \mathrm{ring}), 0.2\rangle) = \text{ⓓ}$; it satisfies the timing constraint $\text{ⓑ} - \text{ⓐ} \leq 1$ trivially, since neither ⓐ nor ⓑ are part of ⓒⓓ. Also, the longer trace $\langle \textsc{snd}(\mathrm{A}, \mathrm{B}, \mathrm{ring}), 0.2\rangle \langle \textsc{rcv}(\mathrm{A}, \mathrm{B}, \mathrm{ring}), 0.9\rangle \langle \textsc{snd}(\mathrm{A}, \mathrm{B}, \mathrm{ring}), 1.1\rangle \langle \textsc{rcv}(\mathrm{A}, \mathrm{B}, \mathrm{ring}), 2.4\rangle$ conforms to the sequence of occurrence specifications ⓐⓑⓒⓓ and satisfies the timing constraint $\text{ⓑ} - \text{ⓐ} \leq 1$. □

4.2 Deriving an Interaction Structure

We now define a function $\mathcal{S}[\![-]\!]\,\Sigma$ that yields an interaction structure (O, R, X, Θ) for an interaction fragment given a context $\Sigma = \{\mathsf{sd}(name_1, T'_1), \ldots, \mathsf{sd}(name_k, T'_k)\}$ of interactions, where the names $name_i$ are pairwise different. In the definition, we proceed recursively by the structure of our abstract syntax of UML 2 interaction fragments, where we always assume that the identifiers of occurrence specifications are globally unique. We write $\mathrm{Min}(O, \preceq)$ and $\mathrm{Max}(O, \preceq)$ for the set of minimal and maximal elements of a partial order $(O, \preceq)$.

Basic Interactions. For a basic interaction $B = (O, \to)$, the occurrence specifications O make up the occurrence specifications component of the resulting interaction structure, and the ordering relation $\to \subseteq O \times O$ yields the ordering constraints:

$$\mathcal{S}[\![(O, \to)]\!]\,\Sigma = (O, \to, \emptyset, \mathrm{true}) \,.$$

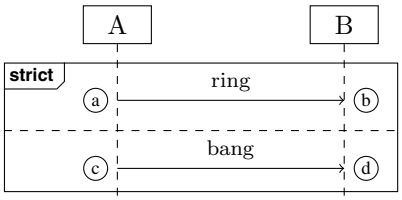
Note that for any interaction structure, we only need to record the skeleton of the partial ordering resulting from the order component which, in general, may reduce the number of pairs to be stored considerably.

Combined Fragments. We give the definitions for strict, seq, par, alt, break, and ref, abbreviating $\mathcal{S}[\![\, T_i \,]\!]\, \Sigma$ by $(O_i, R_i, X_i, \Theta_i)$:

A strict composition $\mathsf{strict}(\, T_1,\, T_2)$ of two timed fragments T_1 and T_2 requires that T_1 has to have completely finished before T_2 starts. For $\mathcal{S}[\![\mathsf{strict}(\, T_1,\, T_2)]\!]\, \Sigma$ we therefore not only take the union (resp. conjunction) of all the components of the interaction structures for T_1 and T_2, but also add the constraint that any occurrence specification from $\mathcal{S}[\![\, T_1 \,]\!]\, \Sigma$ has to occur before any occurrence specification from $\mathcal{S}[\![\, T_2 \,]\!]\, \Sigma$:

$$\begin{aligned}\mathcal{S}[\![\mathsf{strict}(\, T_1,\, T_2)]\!]\, \Sigma = (&O_1 \cup O_2,\\ &R_1 \cup R_2 \cup \{\, o_1 \to o_2 \mid o_1 \in O_1,\ o_2 \in O_2 \,\},\\ &X_1 \cup X_2, \Theta_1 \wedge \Theta_2)\ .\end{aligned}$$

Example 5. Consider the following UML 2 interaction diagram:



The two inner fragments have the interaction structure

$$\begin{aligned}&(\{ⓐ : \textsc{snd}(\mathrm{A}, \mathrm{B}, \mathrm{ring}), ⓑ : \textsc{rcv}(\mathrm{A}, \mathrm{B}, \mathrm{ring})\}, \{ⓐ \to ⓑ\}, \emptyset, \mathrm{true})\ ,\\ &(\{ⓒ : \textsc{snd}(\mathrm{A}, \mathrm{B}, \mathrm{bang}), ⓓ : \textsc{rcv}(\mathrm{A}, \mathrm{B}, \mathrm{bang})\}, \{ⓒ \to ⓓ\}, \emptyset, \mathrm{true})\ .\end{aligned}$$

Combining these strictly adds ⓐ → ⓒ, ⓐ → ⓓ, ⓑ → ⓒ, and ⓑ → ⓓ, thus the interaction structure is

$$\begin{aligned}&(\{ⓐ, ⓑ, ⓒ, ⓓ\},\\ &\ \{ⓐ \to ⓑ, ⓐ \to ⓒ, ⓐ \to ⓓ, ⓑ \to ⓒ, ⓑ \to ⓓ, ⓒ \to ⓓ\}, \emptyset, \mathrm{true})\ ,\end{aligned}$$

where we have omitted the types of the occurrence specifications; taking the skeleton of the partial order specified in the order component, this can be expressed equivalently by

$$(\{ⓐ, ⓑ, ⓒ, ⓓ\}, \{ⓐ \to ⓑ, ⓑ \to ⓒ, ⓒ \to ⓓ\}, \emptyset, \mathrm{true})\ .$$ □

A weak sequential composition $\mathsf{seq}(\, T_1,\, T_2)$ of two timed fragments T_1 and T_2 only requires that T_1 has to have finished before T_2 *lifeline-wise*, i.e., all occurrence specifications of T_1 on some lifeline have to happen before all occurrence specifications of T_2 on the same lifeline. Let us write $o_1 \lesssim o_2$ when o_1 and o_2 are *active* for the same lifeline where a $\textsc{snd}(s, r, m)$ is active for the sender s, and a $\textsc{rcv}(s, r, m)$ is active for the receiver r:

$$\begin{aligned}\mathcal{S}[\![\mathsf{seq}(\, T_1,\, T_2)]\!]\, \Sigma = (&O_1 \cup O_2,\\ &R_1 \cup R_2 \cup \{\, o_1 \to o_2 \mid o_1 \in O_1,\ o_2 \in O_2,\ o_1 \lesssim o_2 \,\},\\ &X_1 \cup X_2, \Theta_1 \wedge \Theta_2)\ .\end{aligned}$$

Example 6. Consider the UML 2 interaction diagram of the previous example but with strict replaced by seq. The two inner fragments have the same interaction structure as before, but the ordering constraints added now are only ⓐ → ⓒ and ⓑ → ⓓ. □

A parallel fragment $\mathsf{par}(T_1, T_2)$ allows for an arbitrary interleaving of the occurrence specifications in T_1 and T_2, as long as the constraints for T_1 and T_2 are satisfied separately; therefore we take the union resp. conjunction of all components of the respective interaction structures:

$$\mathcal{S}[\![\mathsf{par}(T_1, T_2)]\!]\,\Sigma = (O_1 \cup O_2, R_1 \cup R_2, X_1 \cup X_2, \Theta_1 \wedge \Theta_2)\,.$$

An alternative fragment $\mathsf{alt}(T_1, T_2)$ represents a choice of either T_1 or T_2. Here, we express the two possibilities by making the occurrence specifications of T_1 and T_2 mutually exclusive:

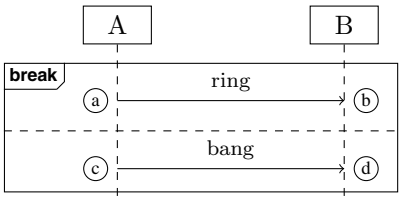
$$\begin{aligned}\mathcal{S}[\![\mathsf{alt}(T_1, T_2)]\!]\,\Sigma = (&O_1 \cup O_2, R_1 \cup R_2,\\ &X_1 \cup X_2 \cup \{o_1 \rightsquigarrow o_2 \mid o_1 \in O_1,\ o_2 \in O_2\} \cup\\ &\qquad \{o_2 \rightsquigarrow o_1 \mid o_1 \in O_1,\ o_2 \in O_2\},\\ &\Theta_1 \wedge \Theta_2)\,.\end{aligned}$$

An example for alt has been given in Ex. 2. However, the representation of the inhibition constraints $X = \{ⓐ \rightsquigarrow ⓒ, ⓐ \rightsquigarrow ⓓ, ⓑ \rightsquigarrow ⓒ, ⓑ \rightsquigarrow ⓓ, ⓒ \rightsquigarrow ⓐ, ⓓ \rightsquigarrow ⓐ, ⓒ \rightsquigarrow ⓑ, ⓓ \rightsquigarrow ⓑ\}$ given there can be reduced to $\{ⓐ \rightsquigarrow ⓒ, ⓐ \rightsquigarrow ⓓ, ⓒ \rightsquigarrow ⓐ, ⓓ \rightsquigarrow ⓐ\}$ using the ordering constraints $R = \{ⓐ \to ⓑ, ⓒ \to ⓓ\}$.

A break fragment $\mathsf{break}(T_1, T_2)$ says that T_1 may be aborted at any time during its execution, and T_2 is performed on abortion. (Note that the UML 2 specification introduces break as a unary interaction operator showing only one interaction fragment as operand; it aborts its enclosing interaction fragment. We prefer to make break binary in order to clarify the two operands.) The translation of break is thus similar to the translation of alt; however, we only require that after T_2 has started, no occurrence specification of T_1 is allowed any more, and if T_1 has finished, no occurrence specification from T_2 is allowed:

$$\begin{aligned}\mathcal{S}[\![\mathsf{break}(T_1, T_2)]\!]\,\Sigma = (&O_1 \cup O_2, R_1 \cup R_2,\\ &X_1 \cup X_2 \cup \{o_2 \rightsquigarrow o_1 \mid o_1 \in O_1,\ o_2 \in O_2\} \cup\\ &\qquad \{o_1 \rightsquigarrow o_2 \mid o_1 \in \mathrm{Max}(O_1, \preceq_{R_1}),\ o_2 \in O_2\},\\ &\Theta_1 \wedge \Theta_2)\,.\end{aligned}$$

Example 7. Consider the following UML 2 interaction diagram:



The resulting inhibition constraints are $\{ⓒ \rightsquigarrow ⓐ, ⓒ \rightsquigarrow ⓑ, ⓓ \rightsquigarrow ⓐ, ⓓ \rightsquigarrow ⓑ, ⓑ \rightsquigarrow ⓒ, ⓑ \rightsquigarrow ⓓ\}$. This is similar to Ex. 3, but $\{ⓑ \rightsquigarrow ⓒ, ⓑ \rightsquigarrow ⓓ\}$ is added. The resulting allowed sequences of occurrence specifications are ⓐⓑ, ⓐⓒⓓ, and ⓒⓓ. □

Finally, a reference fragment $\mathsf{ref}(name)$ amounts to yielding the interaction structure of the interaction fragment T from $\mathsf{sd}(name, T) \in \Sigma$; in order keep all identifiers of occurrence specifications unique, we use a *renaming* ρ with $\rho(o) = name.o$ (where $name.o$ is assumed to be fresh), which we write in postfix notation:

$$\mathcal{S}[\![\mathsf{ref}(name)]\!]\,\Sigma = (O\rho, R\rho, X\rho, \Theta\rho) \quad \text{if } \mathsf{sd}(name, T) \in \Sigma \text{ and } \mathcal{S}[\![T]\!]\,\Sigma = (O, R, X, \Theta)\,.$$

Timing Constraints. For a timed fragment $\mathsf{tmconstr}(T, \Gamma)$, we first reduce each duration constraint $\ell \bowtie d$ in Γ to an expanded form resulting in a timing constraint Θ, characterizing the duration of the interaction fragment T in terms of its occurrence specifications. Then we add this expanded Θ conjunctively to the timing constraints of the interaction structure $\mathcal{S}[\![T]\!]\,\Sigma$:

$$\mathcal{S}[\![\mathsf{tmconstr}(T, \Gamma)]\!]\,\Sigma = (O(\mathcal{S}[\![T]\!]\,\Sigma), R(\mathcal{S}[\![T]\!]\,\Sigma), X(\mathcal{S}[\![T]\!]\,\Sigma), \Theta(\mathcal{S}[\![T]\!]\,\Sigma) \wedge \Theta)\,.$$

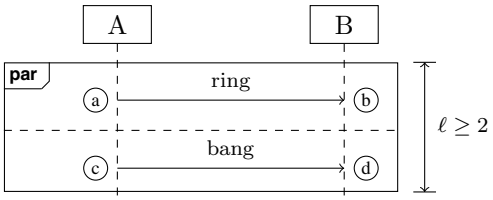
The expansion of an $\ell \bowtie d$ uses the partial order $(O, \preceq) = (O(\mathcal{S}[\![T]\!]\,\Sigma), \preceq_{R(\mathcal{S}[\![T]\!]\,\Sigma)})$ and has to distinguish between the two cases whether $\bowtie \in \{<, \leq\}$ and $\bowtie \in \{\geq, >\}$. In the case of an upper bound where $\bowtie \in \{<, \leq\}$, the expansion is the conjunction of upper bounds between the minimal and the maximal occurrence specifications of T; i.e., all occurrence specifications must happen within time bound d:

$$\bigwedge\{o_2 - o_1 \bowtie d \mid o_2 \in \mathrm{Max}(O, \preceq), o_1 \in \mathrm{Min}(O, \preceq)\}\,.$$

In the case of a lower bound where $\bowtie \in \{\geq, >\}$, the expansion is the disjunction of lower bound between the minimal and the maximal occurrence specifications of T; i.e., the difference in time between some occurrence specifications must be at least d:

$$\bigvee\{o_2 - o_1 \bowtie d \mid o_2 \in \mathrm{Max}(O, \preceq), o_1 \in \mathrm{Min}(O, \preceq)\}\,.$$

Example 8. Consider the following UML 2 interaction diagram:



The duration of the par-fragment shall be at least 2. The minimal elements of the overall interaction structure are $\{ⓐ, ⓒ\}$, the maximal elements $\{ⓑ, ⓓ\}$. The expansion of $\ell \geq 2$ therefore is

$$(ⓑ - ⓐ \geq 2) \vee (ⓑ - ⓒ \geq 2) \vee (ⓓ - ⓐ \geq 2) \vee (ⓓ - ⓒ \geq 2)\,.$$ □

5 Validation

We validate our approach by a prototypical implementation with a large number of test cases, and an extended example, where we check whether given traces comply with an interaction *without* computing all the traces of the interaction first. The transformation of interactions into interaction structures is straightforward and directly follows the definitions given in Sect. 4 above. Algorithm 1 shows how trace prefixes of arbitrary length can be checked for conformance against a given interaction structure. The *successor* function simply takes an ordering or time constraint and extracts the event occurrence with a higher time-stamp. The *choose*-operation is necessary since it is possible that concurrent fragments start with event occurrences with the same signature, as our example shows. In this situation, all alternative paths have to be explored; using Prolog's built in backtracking features allows straightforward handling of this situation. While one can construct abnormal cases where this is indeed occurring, leading to deterioration of performance, it may be argued that this is a modeling mistake, so that most practical scenarios will not suffer from this drawback. Without it, the computational effort of this algorithm is linear in the size of the interaction and the trace.

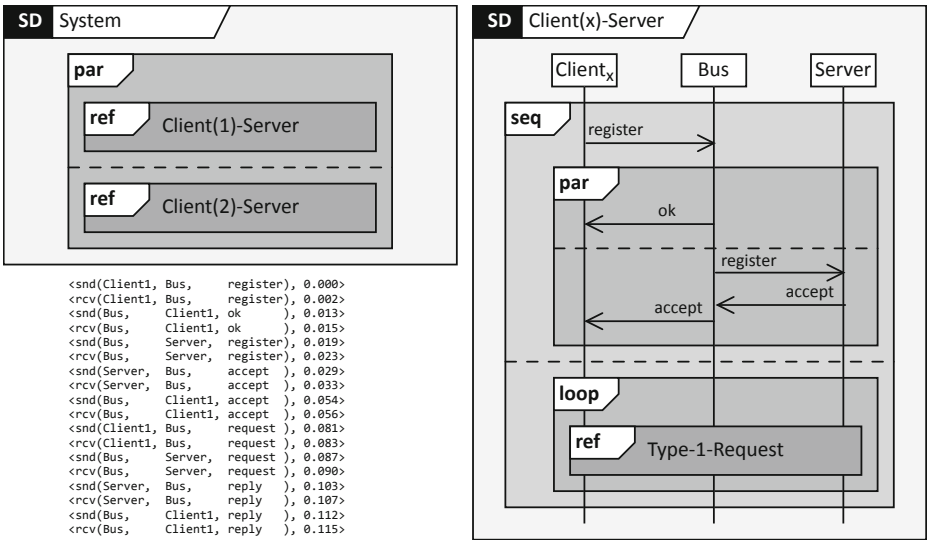
Algorithm 1. Check whether a trace conforms to an interaction structure

Input: an interaction structure IS and a trace of events $e_1 \ \ldots \ e_k$
for $i = 1..k$ **do**
  // Compute unconstrained (i.e., enabled) occurrence specifications
  $U_i \leftarrow O(IS)$;
  for $c \in R(IS) \cup \mathit{constraints}(\Theta(IS))$ **do**
    $U_i \leftarrow U_i \setminus \mathit{successor}(c)$;
  // Choose enabled, conforming occurrence specification, if possible
  $O_i \leftarrow \{o \in U_i \mid e_i$ conforms to $o\}$;
  if $O_i = \emptyset$ **then**
    abort "trace does not conform";
  else
    $o_i \leftarrow$ **choose** O_i;
  // Propagate choice removing irrelevant occurrence specifications and constraints
  $X_i \leftarrow \mathit{conflicting}(o_i, X)$;
  $IS \leftarrow$ remove occurrence specifications $\{o_i\} \cup X_i$ from IS;
  $IS \leftarrow$ remove constraints related to $\{o_i\} \cup X_i$ from IS;
  $IS \leftarrow$ simplify $\Theta(IS)$ with time-stamp $t(e_i)$ for o_i;
// Check all timing constraints
evaluate $\Theta(IS)$;

We have also implemented the symbolic representation and the above algorithm to demonstrate its feasibility. We used SWI-Prolog [15] for this purpose to be able to align the implementation closely with the definitions of this paper. The implementation consists of 5 modules with less than 800 lines of code/1000 clauses, plus a few generic auxiliary libraries. The implementation allows to check interactions for well-formedness,

transform them into timed event structures, expand them to trace sets, and, of course, check traces against interaction structures. Another set of modules (approx. 300 lines of code) defines approx. 100 test cases and run-time measurement scaffold. We have used this implementation to analyze the sample interaction shown in Fig. 1 and Fig. 3. We have defined ten examples and counter-examples of valid traces manually, and checked them for compliance against the interaction, validating that our implementation does indeed truthfully implement our approach. The smallest of these samples for Client(1)-Server is shown in Fig. 3 (bottom, left).

In order to validate the scalability of our approach, we created a loop wrapping a simple elementary interaction (see Fig. 4, top), and checked it against traces of increasing length. In Fig. 4, bottom, we show the length of traces as the x-axis (corresponds to the number of occurrence specifications in the interaction structure), the number of constraints arising from it (y-axis, grey bar chart/graph), and the time used for converting an interaction to an interaction structure (y-axis, red graph). The time to check a trace against an interaction structure was too small to be measured. All measurements are the average of ten runs, to cancel out delays due to garbage collection and similar issues. All measurements were taken on an outdated sub-notebook computer (Intel Core Duo, 1.2GHz, 2GB RAM).




```
<snd(Client1, Bus,     register), 0.000>
<rcv(Client1, Bus,     register), 0.002>
<snd(Bus,     Client1, ok      ), 0.013>
<rcv(Bus,     Client1, ok      ), 0.015>
<snd(Bus,     Server,  register), 0.019>
<rcv(Bus,     Server,  register), 0.023>
<snd(Server,  Bus,     accept  ), 0.029>
<rcv(Server,  Bus,     accept  ), 0.033>
<snd(Bus,     Client1, accept  ), 0.054>
<rcv(Bus,     Client1, accept  ), 0.056>
<snd(Client1, Bus,     request ), 0.081>
<rcv(Client1, Bus,     request ), 0.083>
<snd(Bus,     Server,  request ), 0.087>
<rcv(Bus,     Server,  request ), 0.090>
<snd(Server,  Bus,     reply   ), 0.103>
<rcv(Server,  Bus,     reply   ), 0.107>
<snd(Bus,     Client1, reply   ), 0.112>
<rcv(Bus,     Client1, reply   ), 0.115>
```

Fig. 3. An extended example for validating our approach and implementation. Observe, that the interaction shown in Fig. 1 is re-used.

The measurements clearly show that, with increasing length of loop unrolling and trace length, the number of constraints increases linearly, while the conversion times increase polynomially. Recall, that this translation occurs only once, at model compile time; afterwards, all checks are executed in constant time. Even when including the

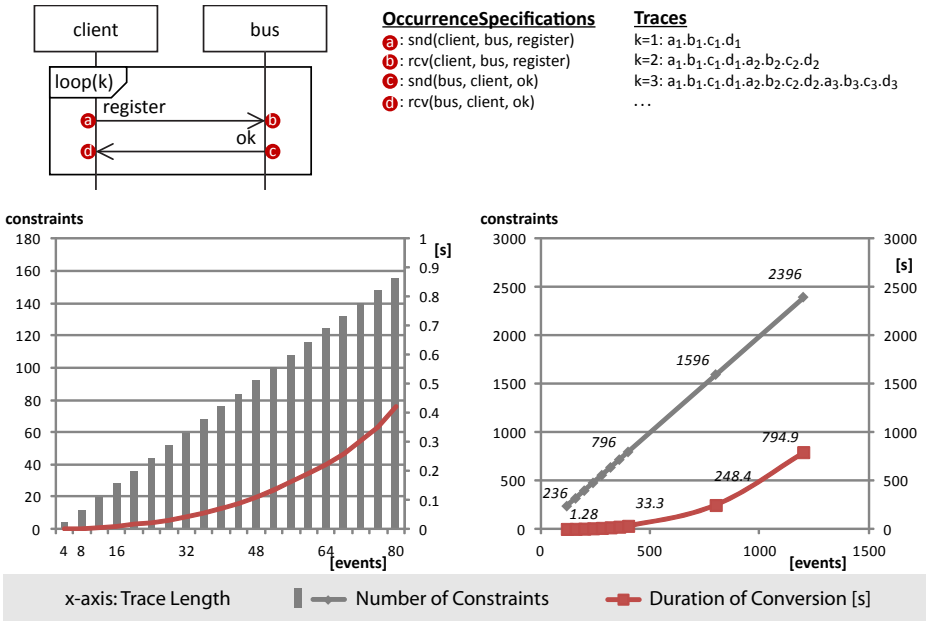


Fig. 4. Measuring the scalability of the implementation: transforming interactions into interaction structures takes linear space and polynomial time. "Constraints" include ordering, timing constraints, and conflicts.

translation time, a naive implementation on weak hardware results in acceptable runtimes: checking a (timed) trace against an interaction takes less than a second for traces of up to 100 events, and about half a minute for traces of around 500 events.

6 Conclusions and Future Work

With interaction structures, we have presented a compact and versatile format for representing the positive trace sets of a UML 2 interaction. Based on asymmetric event structures, interaction structures provide flexible means for specifying alternative scenarios. The format is declarative rather than operational; it relies on constraints for expressing orderings and exclusions and includes timing constraints for expressing real-time requirements in interactions. We have also described a prototypical implementation for translating a UML 2 interaction into an interaction structure and checking the conformance of a trace with the interaction structure.

The approach presented in this paper is the first to cover time for UML 2 interactions in an efficient way, including the major interaction operators. This is essential for practical tool support, which we demonstrate with a proof-of-concept implementation. Previous approaches either suffered from exponential blow-up of the representations, or considered only a much smaller language fragment.

One of the open issues is the inclusion of interaction fragments with empty traces, like opt. This would require a small extension of the notion of interaction structures with "virtual" occurrence specifications that indicate the beginning and ending of an interaction fragment and which can be interpreted as "silent actions" [8], though we would like to minimize their number. A proper integration of the notorious negative behavior specification operators neg and assert is more challenging, where, e.g., modal sequence diagrams may be an interesting approach [6]. We would also like to investigate the use of our algorithm for checking the conformance of a trace to an interaction for run-time verification.

References

1. Baldan, P., Corradini, A., Montanari, U.: Contextual Petri Nets, Asymmetric Event Structures, and Processes. Inf. Comput. 171(1), 1–49 (2001)
2. Dobing, B., Parsons, J.: How UML Is Used. Comm. ACM 49(5), 109–113 (2006)
3. Dobing, B., Parsons, J.: Dimensions of UML Diagram Use: Practitioner Survey and Research Agenda. In: Siau, K., Erickson, J. (eds.) Principle Advancements in Database Management Technologies: New Applications and Frameworks, pp. 271–290. IGI Publishing (2010)
4. Grosu, R., Smolka, S.A.: Safety-Liveness Semantics for UML 2.0 Sequence Diagrams. In: Proc. 5th Conf. Appl. of Concurrency to System Design (ACSD 2005), pp. 6–14. IEEE Computer Society (2005)
5. Hammal, Y.: Branching Time Semantics for UML 2.0 Sequence Diagrams. In: Najm, E., Pradat-Peyre, J.-F., Donzeau-Gouge, V.V. (eds.) FORTE 2006. LNCS, vol. 4229, pp. 259–274. Springer, Heidelberg (2006)
6. Harel, D., Maoz, S.: Assert and Negate Revisited: Modal Semantics for UML Sequence Diagrams. J. Softw. Syst. Model. 7(2), 237–252 (2008)
7. Knapp, A., Wuttke, J.: Model Checking of UML 2.0 Interactions. In: Kühne, T. (ed.) MoDELS 2006. LNCS, vol. 4364, pp. 42–51. Springer, Heidelberg (2007)
8. Küster-Filipe, J.: Modelling Concurrent Interactions. Theo. Comp. Sci. 351(2), 203–220 (2006)
9. Micskei, Z., Waeselynck, H.: The Many Meanings of UML 2 Sequence Diagrams: A Survey. J. Softw. Syst. Model. 10(4), 489–514 (2011)
10. Nayak, A., Samanta, D.: Automatic Test Data Synthesis using UML Sequence Diagrams. J. Obj. Techn. 9(2), 75–104 (2010), http://www.jot.fm/issues/issue201003/article2/
11. Nielsen, M., Plotkin, G., Winskel, G.: Petri Nets, Event Structures and Domains, Part I. Theo. Comp. Sci. 13, 85–108 (1981)
12. Object Management Group: OMG Unified Modeling Language (OMG UML), Superstructure. Version 2.4.1. OMG Document Number: formal/2011-08-06. Tech. rep., Object Management Group (August 2011), http://www.omg.org/spec/UML/2.4.1/
13. Störrle, H.: Assert, Negate and Refinement in UML-2 Interactions. In: Jürjens, J., Rumpe, B., France, R., Fernandey, E.B. (eds.) Proc. Ws. Critical Systems Development with UML. Technical report TUM-I0317. pp. 79–94 (2003)
14. Störrle, H.: Semantics of Interactions in UML 2.0. In: Hosking, J., Cox, P. (eds.) Proc. IEEE Symp. Human Centric Computing Lang. and Env., pp. 129–136. IEEE Computer Society (2003)
15. Wielemaker, J., Schrijvers, T., Triska, M., Lager, T.: SWI-Prolog. Theory and Practice of Logic Programming 12(1-2), 67–96 (2012)